**DATABASES AND DATA STRUCTURE 1**

**MODULE CODE: COMP50004**

**EFFICIENT GOODS LOADING SYSTEM FOR “HANOI ROADWAYS”**

**Individual Coursework:** 23080330

***#Word Count:*** *552*

**Table of Contents**

[**Truck Loading Problems** 3](#_Toc185272367)

[**1.** **Problem Overview** 3](#_Toc185272368)

[**1.1** **Scenario** 3](#_Toc185272369)

[**1.2** **Major Assumption** 3](#_Toc185272370)

[**2.** **Data Structures** 3](#_Toc185272371)

[**3.** **Sample Data** 4](#_Toc185272372)

[**3.1** **Sample Input Data** 4](#_Toc185272373)

[**3.2** **Sample Output Data** 4](#_Toc185272374)

[**4.** **Goods Loading Plan** 4](#_Toc185272375)

[**5.** **Algorithms** 5](#_Toc185272376)

[**6.** **Psuedo-Code** 5](#_Toc185272377)

[**7.** **Coding Implementation** 7](#_Toc185272378)

[**References** 8](#_Toc185272379)

[**Appendix A: Sample Data in Excel Files** 8](#_Toc185272380)

[**Appendix B: Original work without AI usage** 9](#_Toc185272381)

# **Truck Loading Problems**

## **Problem Overview**

### **Scenario**

“Hanoi Roadway” is having trouble loading parcels onto trucks because there is no regulatory system. The laborers are tempted to load the parcels randomly, making it challenging to locate the right parcels during the delivery. As a result, the process seems excessively time-consuming, leading to frustration and unemployment for the laborers.

### **Major Assumption**

The major assumption can be considered as an initial idea that being used for the sake of solving a problem, which defines clearly the boundaries and make it easier in developing a solution without accounting for every possibilities. In this scenario, the major assumption might include:

Parcels characteristics:

* ID: Each parcel has random ID
* Weight: The accepted weight range for parcels is between 1 and 130 kg, weight measurements are accurate before each parcel has loaded onto truck.
* **Others: Some other information that is provided for the parcel includes destination and customers name.**

**Truck assumptions:**

* Truck Type: Single-door trucks are used for loading and unloading
* Truck Capacity: All trucks have the fixed and same capacity which is 500kg

Invoice Generated:

* Each parcel generates a unique invoice containing key details (Parcel ID, destination, weight, costs and customer name).

## **Data Structures**

|  |  |
| --- | --- |
| Data Structure | Purpose |
| Stack | The parcels that are loaded first in a truck will be unloaded last |
| List | Store parcels in each truck, and list of trucks in the system |
| Set | Keep track of destination of each truck |
| Dictionary | Map destinations to distances |
| String | Store text-based information of parcels and trucks’ details |

## **Sample Data**

### **Sample Input Data**

|  |  |  |  |
| --- | --- | --- | --- |
| Parcel ID | Weight | Destination | Customer Name |
| P564814 | 23 | HCMC | Hikaru |
| P708127 | 45 | HCMC | Azuma |
| P132493 | 62 | HCMC | Hwee |
| P987007 | 34 | Dalat | Ngoc |
| P212457 | 123 | Da Nang | Diep |
| P088343 | 78 | Nha Trang | Vinh |
| P371612 | 95 | Nha Trang | Joey |
| P375173 | 92 | HCMC | Gordon |
| P187994 | 5 | Hai Phong | Huy |

Table 1: Sample Input Data

### **Sample Output Data**

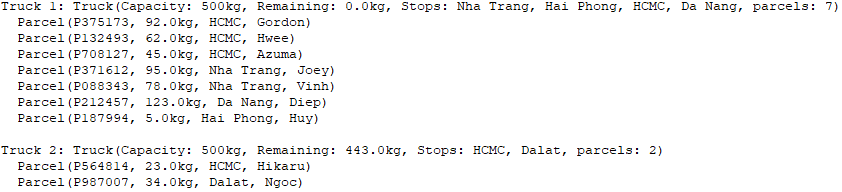


Fig.1: Truck Goods Loading List

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Parcel ID | Destination | Weight (kg) | Distance (km) | Cost ($) | Customer Name |
| P987007 | Dalat | 34 | 250 | 80.5 | Ngoc |

Table 2: Invoice’s details

## **Goods Loading Plan**

Step 1:Gather detailed information on all parcels, including weight, destination city, and customer details.

Step 2: Sort parcels in decreasing order of weight and use a system to allocate parcels to trucks based on parcels’ weight, and truck maximum capacity. Organize the loading sequence such that parcels for the furthest destination city are loaded first (placed at the back of the truck) to simplify unloading at intermediate stops.

Step 3: Stack parcels onto each truck following the organized loading pattern created in Step 2. Ensure parcels for last-to-deliver destinations are loaded first and those for first-to-deliver destinations are loaded last. This arrangement enables laborers to unload parcels easily and efficiently at each stop, reducing confusion and delays.

## **Algorithms**

This scenario will be solved based on the Bin packing problem with the application of Best Fit Decreasing algorithm.

The **Bin Packing Problem** is a well-known optimization challenge that focuses on minimizing the number of bins needed to pack a fixed set of items. The problem is widely applicable in logistics, resource allocation, and storage management, where efficient space utilization is crucial. The **Best Fit Decreasing (BFD)** algorithm is a popular heuristic approach to solving this problem. (Korf, 2002)

The first step of the BFD algorithm involves sorting all items in decreasing order of size. This ensures that larger items are packed first, making better use of the available bin space and reducing the likelihood of inefficiencies. By prioritizing larger items, the algorithm sets a strong foundation for optimal packing.

Once the items are sorted, the algorithm iteratively assigns each item to a bin. Each item is placed in the bin where it leaves the smallest amount of unused space, effectively balancing the load across bins. If no existing bin can accommodate the current item, a new bin is created. (DIMITRIS & CHANLIOGLOU, 2018)

## **Psuedo-Code**

Class Parcel:

Method \_\_init\_\_(parcel\_id, parcel\_weight, parcel\_destination, customer\_name):

Initialize parcel with:

- parcel\_id

- parcel\_weight

- parcel\_destination

- customer\_name

Method \_\_str\_\_():

Return string representing the parcel details

Class Trucks:

Method \_\_init\_\_(truck\_max\_capacity):

Initialize truck with:

- truck\_max\_capacity (maximum capacity)

- remaining\_capacity (same as truck\_max\_capacity)

- parcels (empty list)

- destinations (empty set)

Method truck\_accommodation(parcel):

If remaining capacity >= parcel weight:

Return True

Else:

Return False

Method add\_parcel(parcel):

Parcels.add(parcel)

Sort parcels (key = destination\_distance, reverse = True)

Remaining\_capacity = truck\_max\_capacity – parcel\_weight

Destinations.add(parcel\_destination)

Method \_\_str\_\_():

Return string representing truck's details

Class BinPackingSystem:

Method \_\_init\_\_(truck\_capacity):

Initialize with:

- truck\_capacity (maximum capacity for all trucks)

- trucks (empty list)

Method packing\_parcels(parcels):

Sort\_parcels (key = parcel\_weight, reverse = True)

For each\_parcel in sorted\_parcels:

loading\_truck = None

best\_fitting\_truck\_capacity = truck\_capacity + 1

For each\_truck in trucks:

If truck\_accommodation = True:

truck\_remaining\_capacity\_after\_load = remaining\_capacity – parcel\_weight

If truck\_remaining\_capacity\_after\_load ≤ best\_fitting\_truck\_capacity:

best\_fitting\_truck\_capacity = truck\_remaining\_capacity\_after\_load

loading\_truck = each\_truck

If loading\_truck is found:

Add parcel to fitting\_truck

Else:

Create a new\_truck

Add parcel to the new\_truck

Append the new\_truck to trucks list

Return the list of trucks

Class Application:

Creating GUI class

Destination\_distance\_list{name of city: distance}

Function get\_default\_packages\_from\_excel(file):

Read file

Initialize empty list for default\_packages

For row\_info in file:

Set parcel(parcel\_id, parcel\_weight, parcel\_destination, customer\_name)

Default\_parcels.add(parcel)

Main:

Initialize Tkinter root window

Create an instance of BinPackingApp and start the main loop

## **Coding Implementation**

The program uses Python and leverages external libraries to interact with an Excel file, requiring the user to install Python, Pandas, and OpenPyxl to execute it. It features a GUI with a visual table displaying customer data and provides functionalities such as adding parcels, packing them efficiently, and clearing trucks. The Excel file comes preloaded with default parcel details, and users can add new parcels either manually by editing the file or directly via the GUI.

Github link: <https://github.com/Hikaru2035/Data-Structure-Goods-Loading-System>

# 

# **References**

DIMITRIS, V. & CHANLIOGLOU, F., 2018. *Aparallel approach of Best Fit Decreasing algorithm,* s.l.: s.n.

Korf, R. E., 2002. *A New Algorithm for Optimal Bin Packing,* Los Angeles: AAAI-02 Proceedings.

# **Appendix A: Sample Data in Excel Files**



Table 1: Default input data

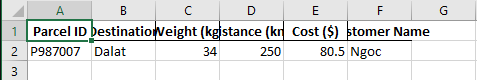


Table 2: Sample invoice output after packing parcel onto trucks

# **Appendix B: Original work without AI usage**

**4. Goods Loading Plan**

Step 1: Collect the dataset from customers' orders.

Step 2: Sort parcels by decreasing in weight and divide parcels into relevant trucks by system to ensure efficient carry of each truck. Then the system will organize the loading pattern list which the furthest city will be on the top of the list to load first.

Step 3: Stack parcels onto each truck in the correct delivery order that has been organized in the aforementioned step, ensuring last-to-deliver parcels are loaded first and first-to-deliver are loaded last according to the provided list in Step 2 which make it easier for laborers to unload parcels when it reaches the destination.

**5. Algorithms**

**Bin Packing Problem** is the classic problem that optimizes the number of bins to pack the constant number of packages.

**Best fit decreasing (BFD) algorithm** is the heuristic algorithm used to solve the bin packing problem.

* The first step of the BFD algorithm is to sort the items by size in decreasing order which ensures larger items are packed first, helping to make it better in use of available space.
* After sorting the items, each item will be sorted into the best bin that leaves the least remaining space after sorting. If no bin can accommodate the item, a new bin will be created.

**7. Coding Implementation**

The coding uses external libraries to connect with the Excel file, so the user must install Python, pandas, and openpyxl to run the program.

The program includes a GUI with a visional table with fields for customer data and other functions such as adding parcels, packing parcels, and clearing trucks. The Excel file contains the default details of some parcels. The user can add parcels manually in the Excel file or directly through the GUI.